

**LAPORAN PRAKTIKUM
PEMROGRAMAN BERORIENTASI OBJEK
MODUL V
INTERFACE & MULTIPLE INHERITANCE**



Oleh:

Ramadhan Wijaya

2211102208

IF-10-M

**S1 TEKNIK INFORMATIKA
INSTITUT TEKNOLOGI TELKOM PURWOKERTO
2024**

I. DASAR TEORI

Multipple inheritance

seperti namanya multiple yang berarti lebih dari satu dan inheritance bisa disebut pewarisan maka modul kali ini membahas tentang pewarisan yang jamak atau lebih dari satu. multiple inheritance memungkinkan suatu kelas mewarisi data maupun method lebih dari satu kelas induk dengan sintaks implements nama_kelas_interfacenya. konstruktor kelas turunan memanggil semua konstruktor kelas induk untuk menginisialisasi data kelas kelas induk. pada saat objek kelas anak diciptakan.

Interface

interface adalah sebuah sebutan yang fungsinya untuk menjembatani dari kelas induk dan anak. didalam interface terdapat method method yang memuat deklarasi struktur method, tanpa detail implementasinya. interface digunakan bila ingin mengaplikasikan suatu method yang spesifik, yaitu tidak diperoleh dari proses pewarisan kelas. interface bersifat disisipkan(embeded) pada program, dan programmer diberi keleluasaan untuk merancang dan mendefinisikan sendiri detail prosesnya. Pada pemanggilan interface di program c++ dan java berbeda berikut adalah penjelasannya :

a. Java

```
public class Xiaomi implements Phone {  
  
    private int volume;  
    private boolean isPowerOn;  
  
    public Xiaomi() {  
        // set volume awal  
        this.volume = 50;  
    }  
}
```

```

@Override
public void powerOn() {
    isPowerOn = true;
    System.out.println("Handphone menyala...");
    System.out.println("Selamat datang di XIAOMI");
    System.out.println("Android version 29");
}

```

Keterangan :

Pada pembuatan class Xiaomi diatas mengimplementasikan interface Phone dengan sintaks implements yang tujuannya agar bisa membuat banyak class lainnya. Coba saja saat phone user akan dimasukan class Samsung maka method method pada Xiaomi tidak cocok dengan method method Samsung dan akan terjadi error. Dari situ fungsi interface yang bisa diunggulkan

b. C++

```

class Terbang
{
public:
    virtual void terbang() = 0; // Fungsi abstrak
};

class Berenang
{
public:
    virtual void berenang() = 0; // Fungsi abstrak
};

class Bebek : public Terbang, public Berenang
{
public:
    void terbang() override
    {
        std::cout << "Bebek terbang!" << std::endl;
    }

    void berenang() override
    {
        std::cout << "Bebek berenang!" << std::endl;
    }
}

```

```
};
int main()
{
    Bebek bebek;
    bebek.terbang();
    bebek.berenang();

    return 0;
}
```

Keterangan :

Pada program c++ lebih simple untuk pewarisan jamak ini jadi pada class yang akan mengimplementasikan lebih dari satu class lainnya hanya perlu menambahkan ":" lalu menambahkan semua kelas yang akan diimplementasikan sebagai kelas induk contohnya :

```
class Bebek : public Terbang, public Berenang
```

Berikut adalah komponen penyusun interfacenya.

Komponen penyusun *interface* adalah sebagai berikut:

Komponen Penyusun	Interface
Tipe data / variabel	Hanya boleh berupa konstanta
Method	Hanya berupa <i>signature</i>, <i>method</i> yang terdapat dalam <i>interface</i> adalah <i>method</i> abstrak
Sintaks <i>method</i>	Tidak perlu membubuhkan <i>modifier</i> <i>abstract</i> pada semua <i>method</i> di dalam kelas

Bila suatu kelas ingin menggunakan *interface*, maka pada deklarasi kelas tersebut ditambahkan keyword **implements** dan nama *interfacenya*. Selanjutnya, *method* dari *interface* tersebut harus disisipkan dan didefinisikan secara detail pada kelas. Yang perlu digarisbawahi adalah bahwa *multiple inheritance* dalam C++ **tidak sama** dengan *interface* dalam Java.

Jadi kesimpulannya adalah pewarisan jamak adalah sebuah sistem untuk memanggil kelas induk yang akan diimplementasikan ke kelas anak/child untuk digunakan seperti template. Kurang lebih sama dengan penjelasan inheritance Tunggal tapi di inheritance jamak ini kelas yang bisa digunakan sebagai class induk bisa lebih dari satu.

II. GUIDED

paketBuku.cpp

```
//Ramadhan Wijaya
//22111102208
//IF-10M
#include <iostream>
#include <string.h>
#include <conio.h>

using namespace std;

class Buku
{
private:
    char judul[100];
    char pengarang[100];
    int jumlah;

public:
    void inisialisasiBuku(char *jdl, char *pngarang, int
jmlh)
    {
        strcpy(judul, jdl);
        strcpy(pengarang, pngarang);
        jumlah = jmlh;
    }
    void infoBuku()
    {
        cout << " Judul :" << judul << endl;
        cout << " Pengarang :" << pengarang << endl;
        cout << " Jumlah buku :" << jumlah << endl;
    }
};

int main(){

    Buku novel, fiksi;
    novel.inisialisasiBuku("Meriam Benteng navarone",
"Alistair Maclean", 12);
    fiksi.inisialisasiBuku("Jurassic park", "Michael
Crichton", 3);
    novel.infoBuku();
    fiksi.infoBuku();
```

```

    getch();
    return 0;
}

```

Output :

```

Judul :Meriam Benteng navarone
Pengarang :Alistair Maclean
Jumlah buku :12
Judul :Jurassic park
Pengarang :Michael Crichton
Jumlah buku :3
PS D:\SEMESTER_3\Praktikum_PBO\MODUL5\C++>

```

Keterangan :

Pada program terlihat tidak ada sistem pewarisan atau inheritance, hanya terdapat class buku dan konstruktor untuk mengisi setiap elemen didalamnya menggunakan strcpy(); untuk tipe data char dan untuk tipe data int secara langsung. Ada juga method infoBuku(); untuk mencetak semua element pada classs buku tersebut.

demoPaket.java

Buku.java

```

package Guided;

/**
 *
 * @author ramadhan wijaya
 * 2211102208
 * if-10m
 */
class Buku {

    String judul, pengarang;
    long hargaBuku;

    public Buku(String judul, String pengarang, long hargaBuku) {

```

```

        this.judul = judul;
        this.pengarang = pengarang;
        this.hargaBuku = hargaBuku;
    }

    public void cetakBuku() {
        System.out.println("\nJudul : " + judul);
        System.out.println("Pengarang : " + pengarang);
        System.out.println("Harga Buku: Rp " + hargaBuku);
        System.out.println();
    }
}

```

Keterangan :

Merupakan program untuk membuat class buku, Dimana didalam class Buku ini berisi elemen elemennya ada judul, pengarang, hargaBuku. Maksud dari pembuatan class Buku ini adalah sebagai class induk dari class class lainnya. Di dalam kelas Buku ini juga terdapat konstruktor untuk mengisi elemen elemen bukunya. Lalu ada prosedur untuk mencetak buku tersebut dengan cetakBuku();

CD.java

```

package Guided;

/**
 *
 * @author ramadhan wijaya
 * 2211102208
 * if-10m
 */
class CD {

    String ukuran;
    long hargaCD;

    public CD(String ukuran, long hargaCD) {
        this.ukuran = ukuran;
        this.hargaCD = hargaCD;
    }

    public long getHargaCD() {
        return hargaCD;
    }
}

```

```

    public void cetakCD() {
        System.out.println("Ukuran CD : " + ukuran);
        System.out.println("Harga CD : Rp " + hargaCD);
        System.out.println();
    }
}

```

Keterangan :

Membuat class CD. Sama seperti class Buku diatas. Ada prosedur cetakCD(); dan konstruktor untuk mengisi elemen elemen pada class CD tersebut juga.

ChildCD.java

```

package Guided;

/**
 *
 * @author ramadhan wijaya
 * 2211102208
 * if-10m
 */
class ChildCD extends CD implements InterfaceCD {

    public ChildCD(String ukuran, long hargaCD) {
        super(ukuran, hargaCD);
    }

}

```

Keterangan :

Class ChildCD adalah class anak/child dari class CD yang dapat diimplementasikan dengan memanggil interfaceCD();. Inilah yang dimaksud interface.

InterfaceCD.java

```

package Guided;

/**
 *
 * @author ramadhan wijaya

```



```

* 2211102208
* if-10-m
*/
interface InterfaceCD {

    void cetakCD();

    long getHargaCD();
}

```

Keterangan :

Class interfaceCD ini dibuat dengan file khusus Namanya Interface.java yang fungsinya khusus untuk membuat class khusus interface. Pada interface hanya terdapat fungsi fungsi dan prosedur yang tak perlu dituliskan lagi badan fungsinya.

Main.java

```

package Guided;

/**
 *
 * @author ramadhan wijaya
 * 2211102208
 * if-10m
 */
class DemoPaket {

    public static void main(String args[]) {
        Paket a = new Paket("Pemrograman Berorientasi Objek",
            "Benyamin Langgu Sinaga", 60000, "700 MB", 50000);
        a.hitungHargaPaket();
        a.cetakPaket();
    }
}

```

Keterangan :

Fungsi utama yang digunakan untuk melakukan run program disini dibuat class DemoPaket dan didalam class ini terdapat fungsi utamanya yaitu membuat tipe data Paket dan menginisialisasinya secara langsung. Lalu ada dua method yang digunakan yaitu method hitungHargaPaket(); dan cetakPaket(); yang Dimana method tersebut terdapat pada class Paket.

Paket.java

```
package Guided;

/**
 *
 * @author ramadhan wijaya
 * 2211102208
 * if-10m
 */
class Paket extends Buku implements InterfaceCD {

    long hargaPaket;
    InterfaceCD interfaceCD;

    public Paket(String judul, String pengarang, long hargaBuku, String ukuran,
long hargaCD) {
        super(judul, pengarang, hargaBuku);
        interfaceCD = new ChildCD(ukuran, hargaCD);
    }

    public void hitungHargaPaket() {
        hargaPaket = super.hargaBuku + getHargaCD();
    }

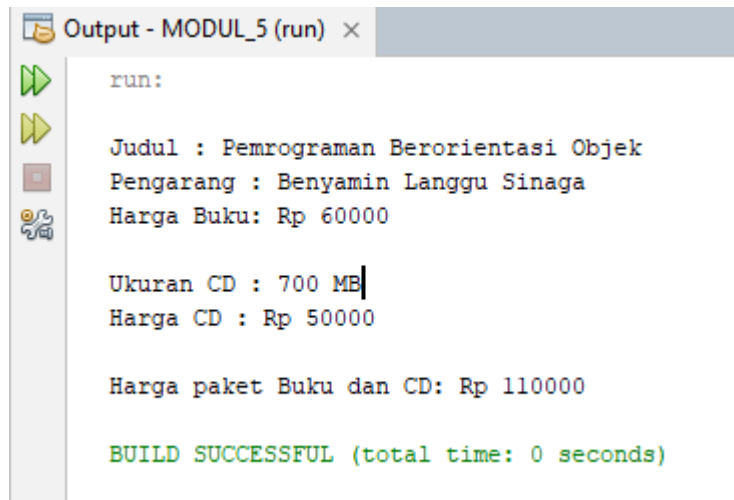
    public void cetakCD() {
        interfaceCD.cetakCD();
    }

    public long getHargaCD() {
        return (interfaceCD.getHargaCD());
    }

    public void cetakPaket() {
        super.cetakBuku();
        cetakCD();
        System.out.println("Harga paket Buku dan CD: Rp " + hargaPaket
            + "\n");
    }
}
```

Keterangan :

Output



```
run:
Judul : Pemrograman Berorientasi Objek
Pengarang : Benyamin Langgu Sinaga
Harga Buku: Rp 60000

Ukuran CD : 700 MB
Harga CD : Rp 50000

Harga paket Buku dan CD: Rp 110000

BUILD SUCCESSFUL (total time: 0 seconds)
```

Keterangan :

pada program utama kita buat tipe data paket yang baru dengan nama "a". setelah kita inisialisasi secara statis, program selanjutnya adalah mengeksekusi method a.cetakPaket(); cetak paket berasal dari file Paket. didalam cetakPaket(); terdapat method cetakBuku(); yang berasal dari class Buku dan print "harga paket buku dan CD : " dan menampilkan variable hargaPaket dimana harga paket berasal dari method HitungHargaPaket(); dimana rumus hargaPaket adalah super.HargaBuku + getHargaCD();. getHargaCD() berada di class CD dan super.hargaBuku adalah variable dari class Buku.

begitulah kuranglebih jalannya program ketika di run dan mengeluarkan output seperti diatas

Untuk dibahas dalam Laporan :

“Multiple Interface Inheritance BUKAN Multiple Implementation Inheritance”

Apakah Anda setuju dengan pernyataan di atas? Jika iya, jelaskan mengapa dan apa perbedaannya! Dan jika tidak, berikan alasannya!

Jawab :

Ya, setuju kalo interface bukan multiple implementation inheritance, karena dari pengimplementasian kedua sistem ini berbeda kalo multiple inheritance itu contohnya seperti ini :

```
class Bebek : public Terbang, public Berenang
```

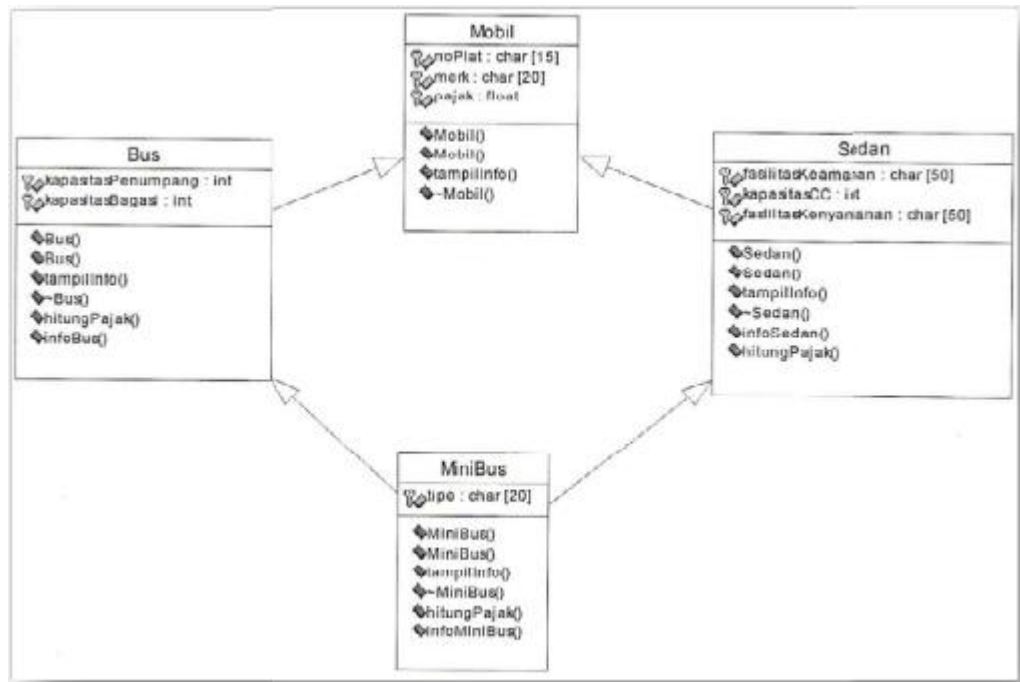
Sedangkan interface pada java seperti ini :

```
class ChildCD extends CD implements InterfaceCD {}
```

Dari segi kelemahan juga ada untuk multiple inheritance memungkinkan terjadinya tipe data yang ambigu sedangkan interface bisa menyesuaikan kelas yang kita inginkan

III. UNGUIDED

perhatikan object relationship diagram pada gambar dibawah ini. anda diminta untuk mengimplementasikan kasus pewarisan jamak pada diagram tersebut.



Ketentuan:

1. Buatlah kelas Mobil dengan ketentuan sebagai berikut (20%) :
 - a. memiliki atribut : noPlat, merk, pajak
 - b. memiliki default konstruktor dan konstruktor buatan
 - c. memiliki method tampilInfo(), untuk menampilkan informasi Mobil
2. Buatlah kelas Bus dengan ketentuan sebagai berikut (20%) :
 - a. turunan dari kelas Mobil
 - b. memiliki atribut : kapasitasPenumpang, kapasitasBagasi
 - c. memiliki default konstruktor dan konstruktor buatan
 - d. memiliki method infoBus(), untuk menampilkan informasi Bus (kapasitasPenumpang, kapasitasBagasi)
 - e. memiliki method tampilInfo(), untuk menampilkan informasi keseluruhan dari Bus (memanggil method tampilInfo () dari kelas Mobil dan method infoBus())
 - f. memiliki method float hitungPajak(), untuk mengembalikan perhitungan besar pajak dengan rumus : $pajak + (pajak * kapasitaspenumpang * kapasitasBagasi * 0.00005)$

3. Buatlah kelas Sedan dengan ketentuan sebagai berikut (20%) :
- a. turunan dari kelas Mobil
 - b. memiliki *atribut* : fasilitasKeamanan, kapasitasCC, fasilitasKenyamanan
 - c. memiliki *default* konstruktor, dan konstruktor bentukan
 - d. memiliki *method* infoSedan(), untuk menampilkan informasi Sedan (fasilitasKeamanan, kapasitasCC, fasilitasKenyamanan)
 - e. memiliki *method* tampilInfo(), untuk menampilkan informasi keseluruhan dari Bus (memanggil *method* tampilInfo() dari kelas Mobil dan *method* infoSedan())
 - f. memiliki *method* float hitungPajak(), untuk mengembalikan perhitungan besar pajak dengan rumus: $pajak + (pajak + (pajak * kapasitasCC * 0.00005))$
4. Buatlah kelas MiniBus dengan ketentuan sebagai berikut (40%) :
- a. turunan dari kelas Sedan, Bus
 - b. memiliki *atribut* : tipe
 - c. memiliki *default* konstruktor dan konstruktor bentukan
 - d. memiliki *method* infoMiniBus(), untuk menampilkan informasi MiniBus (jika tipe adalah "Pribadi" maka tampilkan "Tipe MiniBus : Pribadi, digunakan sebagai kendaraan pribadi", sedangkan jika tipe adalah "Wagon" maka tampilkan "Tipe MiniBus : Wagon digunakan sebagai kendaraan angkut/travel")
 - e. memiliki *method* tampilInfo(), untuk menampilkan informasi keseluruhan dari miniBus (memanggil *method* tampilInfo() dari kelas Sedan, *method* infoBus() dari kelas Bus, dan *method* infoMiniBus ())
 - f. memiliki *method* float hitungPajak(), untuk mengembalikan perhitungan besar pajak dengan rumus :
 - Jika tipe adalah "Pribadi", maka rumusnya : $(Sedan::hitungPajak () * 0.05) + (Bus::hitungPajak () * 0.03)$
 - Jika tipe adalah "Wagon", maka rumusnya : $(Sedan::hitungPajak () * 0.03) + (Bus::hitungPajak () * 0.05)$

Package Unguided

Bus.java

```
/**
 * @author ramadhan wijaya
 * 2211102208
 * if-10m
 */
class Bus extends Mobil {
    int kapasitasBagasi, kapasitasPenumpang;
```

```

    public Bus(int kapasitasPenumpang, int kapasitasBagasi,String noPlat,
String merk, int pajak) {
        super(noPlat, merk, pajak);
        this.kapasitasPenumpang = kapasitasPenumpang;
        this.kapasitasBagasi = kapasitasBagasi;
    }

    public void infoBus(){
        System.out.println("\n kapasitas penumpang adalah
"+kapasitasPenumpang+" orang");
        System.out.println("\n kapasitas Bagasi adalah "+kapasitasBagasi+"
KG");
    }

    public float hitungPajak(){
        float besarPajak = (float) (pajak + (pajak *
kapasitasPenumpang * kapasitasBagasi *0.00005));
        return besarPajak;
    }

    public void tampilInfo(){
        System.out.println("\n plat mobil : " + noPlat);
        System.out.println("\n merk : " + merk);
        System.out.println("\n pajak : Rp. " + hitungPajak());
        infoBus();
    }
}

```

Keterangan :

Pembuatan class Bus yang merupakan kelas child dari class mobil.
 Didalamnya ada konstruktor dan method tampilInfo() dan hitungPajak() dan sebagainya.

ChildBus.java

```

/**
 * @author ramadhan wijaya
 * 2211102208
 * if-10m
 */
class ChildBus extends Bus implements InterfaceBus {

```

```

    public ChildBus(int kapasitasPenumpang, int kapasitasBagasi, String
noPlat, String merk, int pajak) {
        super(kapasitasPenumpang, kapasitasBagasi,noPlat, merk, pajak);
    }
}

```

Keterangan :

Child dari class Bus untuk mempermudah penggunaan class interface nantinya di interfaceBus.java

ChildSedan.java

```

/**
 * @author ramadhan wijaya
 * 2211102208
 * if-10m
 */
class ChildSedan extends Sedan implements InterfaceSedan {

    public ChildSedan(String fasilitasKeamanan, String
fasilitasKenyamanan,int kapasitasCC, String noPlat, String merk, int
pajak) {
        super(fasilitasKeamanan,fasilitasKenyamanan,kapasitasCC,noPlat,
merk, pajak);
    }
}

```

Keterangan :

Fungsinya sama seperti ChildBus untuk mempermudah penggunaan interfaceSedan nantinya

InterfaceBus.java

```

/**
 * @author ramadhan wijaya
 * 2211102208

```



```

* if-10m
*/
public interface InterfaceBus {

    public abstract void infoBus();
    public abstract void tampilInfo();
    public abstract float hitungPajak();
}

```

Keterangan :

interfaceBus digunakan pada saat pembuatan class MiniBus nantinya karena menyesuaikan ketentuan pada soal. Isi interface hanya method method yang dapat diaplikasikan nantinya saat kita mengimplementasikan ke salah satu kelas. Pada interface juga tidak ada variable kecuali variable yang constant.

InterfaceSedan.java

```

/**
 * @author ramadhan wijaya
 * 2211102208
 * if-10m
 */
public interface InterfaceSedan {

    public abstract void infoSedan();
    public abstract void tampilInfo();
    public abstract float hitungPajak();
}

```

Keterangan :

sama seperti interfaceBus.java hanya saja ini untuk interfaceSedan.

Minibus.java

```

/**

```

```

* @author ramadhan wijaya
* 2211102208
* if-10m
*/
class MiniBus implements InterfaceBus,InterfaceSedan {
    float besarPajak;
    String tipe, noPlat, merk;
    InterfaceBus interfacebus;
    InterfaceSedan interfacesedan;

    public MiniBus(String tipe,int kapasitasPenumpang, int
    kapasitasBagasi,String fasilitasKeamanan,String fasilitasKenyamanan, int
    kapasitasCC, String noPlat, String merk, int pajak){
        this.tipe = tipe;
        this.noPlat = noPlat;
        this.merk = merk;

        interfacebus = new
    ChildBus(kapasitasPenumpang,kapasitasBagasi,noPlat,merk,pajak);
        interfacesedan = new
    ChildSedan(fasilitasKeamanan,fasilitasKenyamanan,kapasitasCC,noPlat,
    merk,pajak);
    }

    public void infoMiniBus() {
        System.out.println("\n plat mobil : " + this.noPlat);
        System.out.println("\n merk : " + this.merk);

        if (tipe.equalsIgnoreCase("Pribadi")) {
            System.out.println("\n Tipe MiniBus : Pribadi, digunakan sebagai
kendaraan pribadi");
        } else if (tipe.equalsIgnoreCase("Wagon")) {
            System.out.println("\n Tipe MiniBus : Wagon, digunakan sebagai
kendaraan angkut/travel");
        } else {
            System.out.println("\n Tipe MiniBus tidak valid");
        }
    }

    @Override
    public void infoBus(){
        interfacebus.infoBus();
    }
}

```

```

@Override
public void infoSedan() {
    interfacesedan.infoSedan();
}

@Override
public void tampilInfo(){

    infoMiniBus();
    infoSedan();
    infoBus();
    this.hitungPajak();

}

@Override
public float hitungPajak() {
    if (tipe.equalsIgnoreCase("Pribadi")) {
        besarPajak = interfacesedan.hitungPajak() * 0.05f +
interfacebus.hitungPajak() * 0.03f;
        System.out.println("\n pajak dari kendaraan ini : "+besarPajak+"
Rp");
    } else if (tipe.equalsIgnoreCase("Wagon")) {
        besarPajak = interfacesedan.hitungPajak() * 0.03f +
interfacebus.hitungPajak() * 0.05f;
        System.out.println("\n pajak dari kendaraan ini : "+besarPajak+"
Rp");
    } else {
        System.out.println(" Tipe MiniBus tidak valid");
    }
    return besarPajak;
}
}

```

Keterangan :

Class ini mengimplementasikan interface bus dan sedan, yang Dimana class bus dan sedan adalah class child dari class mobil, jadi di dalam class MiniBus ini terdapat semua variable dari class class induknya.

Karena class MiniBus ini mengimplementasikan interface maka di dalam class ini wajib menaruh method method yang sudah tersedia di kedua interfacenya. Tapi karena ada method yang sama maka dapat di overriding sekali saja.

Sifat dari method dari interface ini tidak mewajibkan sama seperti method pada kelas induknya jadi nama methodnya harus sama, sedangkan badan methodnya boleh berbeda, nah method yang badan fungsinya yang berbeda ini yang disebut overloading.

Mobil.java

```
/**
 * @author ramadhan wijaya
 * 2211102208
 * if-10m
 */
public class Mobil {
    String noPlat,merk;
    float pajak;

    public Mobil(String noPlat, String merk, int pajak){
        this.noPlat = noPlat;
        this.merk = merk;
        this.pajak = pajak;
    }

    public void tampilInfo(){
        System.out.println("\n plat mobil : " + noPlat);
        System.out.println("\n merk mobil : " + merk);
        System.out.println("\n pajak mobil : " + pajak + "\n");
    }
}
```

Keterangan :

Class Mobil ini merupakan kelas induk paling atas dari ketiga class yang ada pada skema soal. Disini class mobil hanya memiliki konstruktor dan method tampil info saja.

Sedan.java

```
/**
```

```

* @author ramadhan wijaya
* 2211102208
* if-10m
*/
class Sedan extends Mobil {

    String fasilitasKeamanan,fasilitasKenyamanan;
    int kapasitasCC;
    public Sedan(String fasilitasKeamanan,String fasilitasKenyamanan,int
kapasitasCC, String noPlat, String merk, int pajak) {
        super(noPlat, merk, pajak);
        this.fasilitasKeamanan = fasilitasKeamanan;
        this.fasilitasKenyamanan = fasilitasKenyamanan;
        this.kapasitasCC = kapasitasCC;
    }

    public void infoSedan(){
        System.out.println("\n Fasilitas Keamanan adalah
"+fasilitasKeamanan);
        System.out.println("\n Fasilitas Kenyamanan adalah
"+fasilitasKenyamanan);
        System.out.println("\n Tenaga : "+kapasitasCC+ "cc");
    }

    public float hitungPajak(){
        float besarPajak = (float) (pajak + (pajak *kapasitasCC *0.00005));
        return besarPajak;
    }

    public void tampilInfo(){
        System.out.println("\n plat mobil : " + noPlat);
        System.out.println("\n merk : " + merk);
        System.out.println("\n pajak : Rp. " + this.hitungPajak());
        infoSedan();
    }

}

```

Keterangan :

Class Sedan merupakan Class yang sama seperti class Bus karena sama sama child dari class mobil. Dalam class Sedan terdapat konstruktor, method infosedan(), hitungPajak karean pada class sedan ini harga pajaknya memiliki rumus sesuai ketentuan, dan tampilinfo() untuk menampilkan semua info dari semua elemen class sedan ini.

Main.java

```
/**
 *
 * @author ramadhan wijaya
 * 2211102208
 * if-10m
 */
public class main {
    public static void main(String args[]) {
        MiniBus minibus = new MiniBus("wagon",100, 200,"smart
lock","kursi sutra",2000,"M21 2211102208 MK","TOYOTA",4000);
        Sedan sedan = new Sedan("kunci gembok","kursi lapis emas",1500,
"B 3432STR MYI","MITSUBISHI",2000);
        Bus bus = new Bus(70,200,"P 7471 VW","GRAND MAX",1000);
        Mobil mobil = new Mobil("R 554 JX","AVANZA",1000000);

        System.out.println("\n\n\n==DISPLAY DATA MOBIL
MINIBUS==");
        minibus.tampilInfo();

        System.out.println("\n\n\n==DISPLAY DATA MOBIL
SEDAN==");
        sedan.tampilInfo();

        System.out.println("\n\n\n==DISPLAY DATA MOBIL BUS==");
        bus.tampilInfo();

        System.out.println("\n\n\n==DISPLAY DATA MOBIL==");
        mobil.tampilInfo();

    }
}
```

Ouput:

```
Output - MODUL_5 (run) x
run:

==DISPLAY DATA MOBIL MINIBUS==

plat mobil : M21 2211102208 MK

merk : TOYOTA

Tipe MiniBus : Wagon, digunakan sebagai kendaraan angkut/travel

Fasilitas Keamanan adalah smart lock

Fasilitas Kenyamanan adalah kursi sutra

Tenaga : 2000cc

kapasitas penumpang adalah 100 orang

kapasitas Bagasi adalah 200 KG

pajak dari kendaraan ini : 532.0 Rp

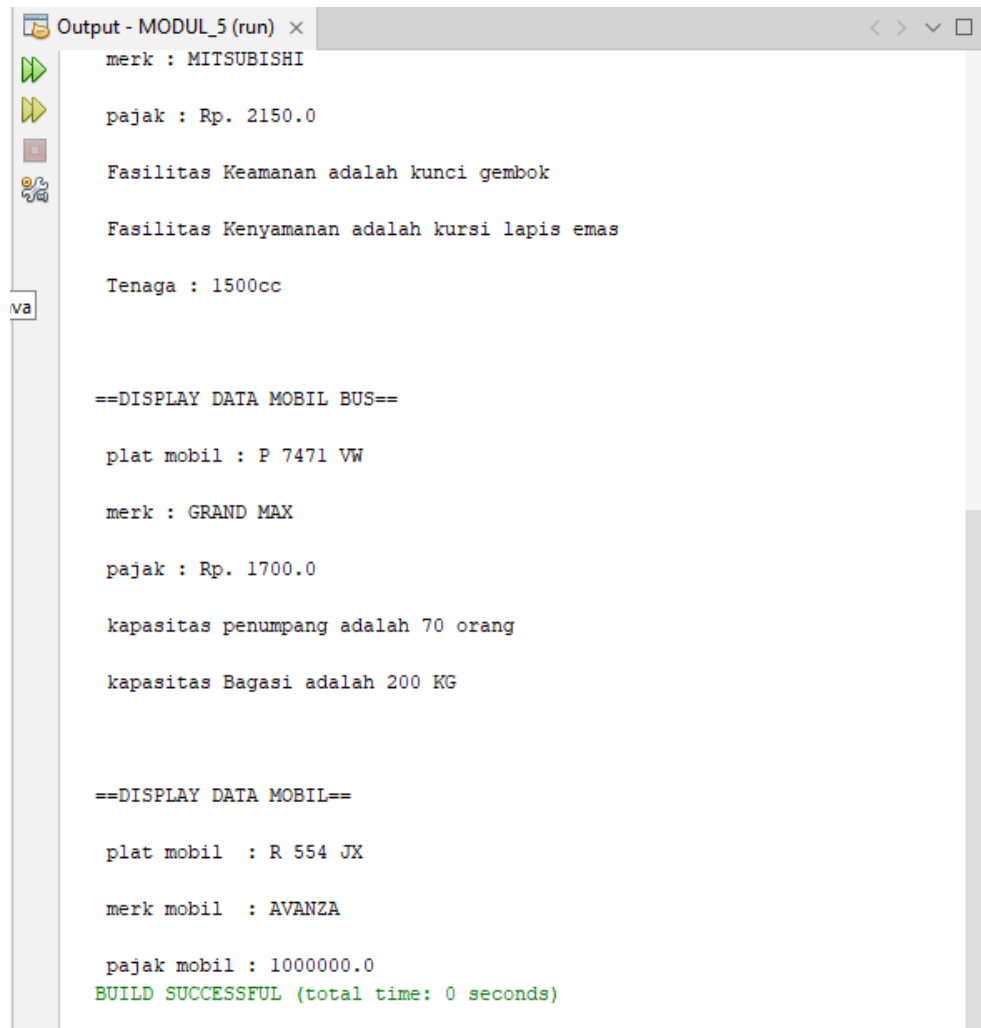
==DISPLAY DATA MOBIL SEDAN==

plat mobil : B 3432STR MYI

merk : MITSUBISHI

pajak : Rp. 2150.0

Fasilitas Keamanan adalah kunci gembok
```



```
merk : MITSUBISHI

pajak : Rp. 2150.0

Fasilitas Keamanan adalah kunci gembok

Fasilitas Kenyamanan adalah kursi lapis emas

Tenaga : 1500cc

==DISPLAY DATA MOBIL BUS==

plat mobil : P 7471 VW

merk : GRAND MAX

pajak : Rp. 1700.0

kapasitas penumpang adalah 70 orang

kapasitas Bagasi adalah 200 KG

==DISPLAY DATA MOBIL==

plat mobil  : R 554 JX

merk mobil  : AVANZA

pajak mobil : 1000000.0
BUILD SUCCESSFUL (total time: 0 seconds)
```

Keterangan :

Adalah program utama yang fungsinya membuat variable dari class class baru seperti Mobil, Minibus, Sedan, dan Bus, sekaligus mengisi parameter sesuai urutan parameter dari konstruktornya. Terakhir dicetak menggunakan method tampilInfo() Dimana method ini memiliki isi yang berbeda beda dari satu sama lain karena method ini dimpor dari Classnya langsung jadi badan methodnya menyesuaikan dari classnya sendiri sendiri.